BAB II

LANDASAN TEORI

2.1 Speaker Recognition

Berbicara adalah cara berkomunikasi yang paling alami antar manusia. Dalam bidang interaksi manusia dengan mesin, sistem *speaker recognition* bertujuan untuk mengenali identitas dari manusia berdasarkan suara ucapan (Hennebert, 2009). Proses otentikasi yang dilakukan dalam sistem *speaker recognition* dapat dijabarkan menjadi proses yang menjamin kebenaran dari identitas yang dikenali tersebut. Proses otentikasi yang dilakukan disebut sebagai otentikasi *closed-set* apabila suara yang diolah selalu merupakan suara dari manusia yang terdaftar dalam sistem, sedangkan otentikasi *open-set* dapat melakukan klasifikasi untuk suara yang tidak terdaftar (Hennebert, 2009).

Otentikasi entitas adalah suatu metode atau proses yang digunakan untuk memberikan jaminan bahwa suatu entitas adalah benar-benar entitas yang diakui. Pada dasarnya, otentikasi dilakukan dengan cara menyediakan suatu tantangan yang hanya dapat diselesaikan atau ditanggapi dengan benar oleh entitas yang dimaksud. Proses ini memberikan keyakinan pada satu pihak akan identitas dari pihak kedua yang terlibat (Menezes, Oorschot, & Vanstone, 1997).

Manfaat utama dari proses otentikasi entitas adalah untuk memberikan kontrol akses terhadap suatu sumber daya yang terhubung dengan identitas tertentu, seperti akses perangkat komputer. Otentikasi dapat melekatkan akses sumber daya terhadap suatu entitas untuk tujuan penagihan, atau dapat mencegah terjadinya pencurian identitas yang dapat memberikan hak akses sumber daya terhadap entitas yang tidak berhak (Menezes, Oorschot, & Vanstone, 1997).

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

Otentikasi entitas dapat dilakukan dengan berbagai cara, dan menurut (Menezes, Oorschot, & Vanstone, 1997) dikategorikan menjadi tiga sesuai dengan basis keamanan yang digunakan.

1. Sesuatu yang diketahui atau *knowledge-based*, yaitu suatu rahasia yang hanya diketahui oleh entitas tersebut, seperti kata sandi dan *Personal Identification Number* (PIN).
2. Sesuatu yang dimiliki atau *token-based*, yaitu benda fisik yang secara khusus hanya dimiliki entitas tersebut, seperti kartu kredit dan paspor.
3. Sesuatu yang melekat atau biometrik, yaitu karakteristik fisik (seperti sidik jari, suara, dan retina) atau perilaku (seperti tanda tangan, cara berbicara, dan kebiasaan mengetik) manusia yang unik dan hanya dimiliki oleh entitas tersebut.

2.2 Biometrik Suara

Otentikasi *token-based* dan *knowledge-based* merupakan dua teknik otentikasi tradisional yang umum digunakan dan memiliki kekurangan, seperti kehilangan *token* atau melupakan kata sandi (Jain, Hong, & Pankanti, 2000). Hal ini diakibatkan pendekatan kedua metode otentikasi tersebut yang tidak mampu membedakan antara suatu entitas berwenang dengan entitas lain yang mendapatkan *token* atau *knowledge* tersebut.

Otentikasi biometrik adalah teknik otentikasi entitas berdasarkan karakteristik fisik atau perilaku entitas (pengenal biometrik) yang dapat dibedakan dengan jelas (Jain, Hong, & Pankanti, 2000). Pengenal biometrik lebih dapat diandalkan untuk melakukan otentikasi karena karakteristik fisik atau perilaku yang digunakan bersifat istimewa dan khusus antar entitas. Selain itu, (Jain, Hong, &

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

Pankanti, 2000) menuturkan bahwa pengukuran biometrik yang tidak memerlukan sentuhan, seperti suara ucapan atau retina, dipandang lebih mudah digunakan.

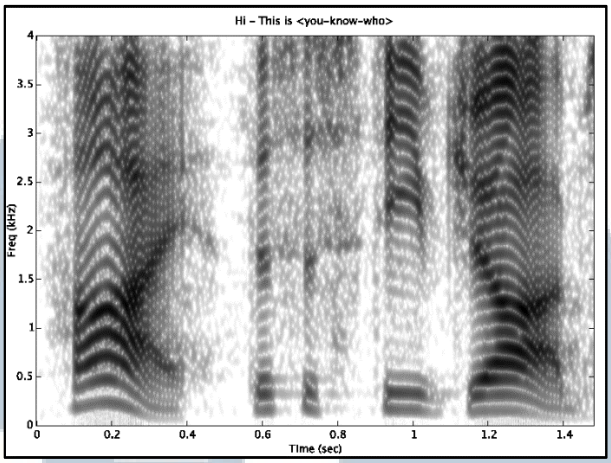
Salah satu biometrik manusia yang dapat digunakan untuk sistem otentikasi entitas adalah suara. Keunikan karakteristik suara manusia dipengaruhi oleh variasi pada ukuran dan bentuk dari saluran vokal, mulut, rongga hidung, dan bibir yang menciptakan suara (Jain, Hong, & Pankanti, 2000). Proses otentikasi menggunakan suara manusia disebut juga *speaker recognition*.

Walaupun sistem *speaker recognition* memiliki tingkat penerimaan pengguna yang tinggi, namun karakteristik berbasis ucapan manusia rentan terhadap beberapa faktor, seperti kebisingan atau *noise* saat suara direkam, kondisi fisik individu seperti flu atau radang tenggorokan, dan kondisi psikis individu (Jain, Hong, & Pankanti, 2000). Dikarenakan hal tersebut, variabilitas pada suara individu menjadi tinggi, dan tingkat akurasi sistem *speaker recognition* pada umumnya dinilai kurang akurat dibandingkan sistem yang menggunakan biometrik lain (Hennebert, 2009).

2.3 Spektrogram

Spektrogram adalah sebuah representasi dari perubahan konten frekuensi pada suatu sinyal suara seiring dengan waktu. Waktu ditampilkan pada sumbu x, frekuensi ditampilkan pada sumbu y, dan amplitudo atau energi yang dikandung pada suatu frekuensi dan waktu ditampilkan dengan intensitas warna. Secara konvensional, warna hitam digunakan untuk menampilkan energi paling tinggi, sedangkan warna putih untuk menampilkan energi paling rendah (UCL, 2003), seperti pada Gambar 2.1.

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 2.1 Representasi Spektrogram dari Suara Manusia (Smith, 2007)

Pada dasarnya, seluruh sinyal terdiri atas banyak gelombang sinus, termasuk sinyal suara. Setiap konten frekuensi pada sinyal dapat ditemukan pada salah satu dari gelombang sinus tersebut. Gelombang ini memiliki karakteristik yang ditentukan oleh frekuensi, amplitudo, dan fase awal gelombang (Seppänen, 1999). Untuk mengubah sinyal suara menjadi gelombang-gelombang sinus yang membangun sinyal tersebut, digunakan proses yang disebut *Discrete Fourier Transform. Discrete fourier transform* menguraikan suatu sinyal menjadi sejumlah gelombang sinusoid statis, sehingga tidak dapat menunjukkan gerak perubahan pada konten frekuensi. Oleh karena itu, *discrete fourier transform* hanya dapat digunakan untuk menguraikan konten frekuensi pada waktu tertentu (Seppänen, 1999).

Suatu spektrogram terbentuk atas sebuah rangkaian spektrum yang disusun berurutan sesuai dengan waktu dan memiliki representasi amplitudo menggunakan intensitas warna (UCL, 2003). Spektrum yang menyusun sebuah spektrogram merupakan sebuah potret konten frekuensi dari sinyal pada waktu tertentu. Spektrum dapat dihasilkan dari suatu sinyal suara dengan membagi-bagi sinyal ke dalam bagian-bagian dengan durasi pendek yang disebut *window*, lalu melakukan

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

discrete fourier transform terhadap bagian tersebut. Bagian-bagian dari sinyal yang telah ditransformasi dengan *discrete fourier transform* menjadi spektrum kemudian disusun secara berurutan untuk membentuk sumbu waktu (Smith, 2007).

Spektrogram merupakan representasi informasi suara yang penting karena bagian dari pendengaran manusia yaitu koklea dan telinga dalam yang bekerja seperti membaca spektrogram (Smith, 2007). Spektrogram merupakan alat dasar pada bidang analisis suara, dan telah digunakan secara luas pada bidang tersebut. Spektrogram juga memiliki kemampuan representasi sinyal suara yang sangat baik, berarti representasi spektrogram dari suatu sinyal memiliki kesamaan yang tinggi dengan sinyal sumbernya (Smith, 2007).

Spektrogram yang dihasilkan memiliki parameter yang dapat diubah, seperti tinggi spektrogram, pixel per detik, fungsi *window* yang digunakan, dan warna (Bagwell, 2014). Pada bidang analisis suara manusia, warna yang digunakan adalah warna tunggal *grayscale* (UCL, 2003), dan *window* yang digunakan pada umumnya diatur untuk memiliki lebar 1 pixel per 20 milidetik (Smith, 2007). Spektrogram yang menggunakan frekuensi di bawah 5.5 KHz menghasilkan vektor dengan 128 elemen angka (Harutyunyan & Khachatrian, 2016). Rentang frekuensi ini dapat digunakan dalam analisis suara manusia karena rentang frekuensi yang digunakan pada saluran telepon, yaitu 300–3400 Hz, dinilai cukup untuk mengenal suara pembicara dan suasana hati pembicara tersebut (Cioara & Valentine, 2012).

2.4 Kecerdasan Buatan

Kecerdasan buatan adalah bidang penelitian mengenai perancangan dan pembangunan sistem kognitif. Bidang penelitian kecerdasan buatan memiliki tujuan untuk merancang kecerdasan komputer setinggi mungkin, atau menyerupai

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

kemampuan dan cara berpikir makhluk hidup sedekat mungkin (Goel & Davies, 2011). Kecerdasan buatan terdiri atas tiga bagian dasar, yaitu aksi, persepsi, dan pengartian. Suatu sistem dengan kecerdasan buatan menerima masukan sebagai persepsi terhadap lingkungannya, mengakses basis pengetahuannya untuk mengartikan persepsi tersebut dan menentukan aksi terbaik, lalu melakukan aksi tersebut (Goel & Davies, 2011).

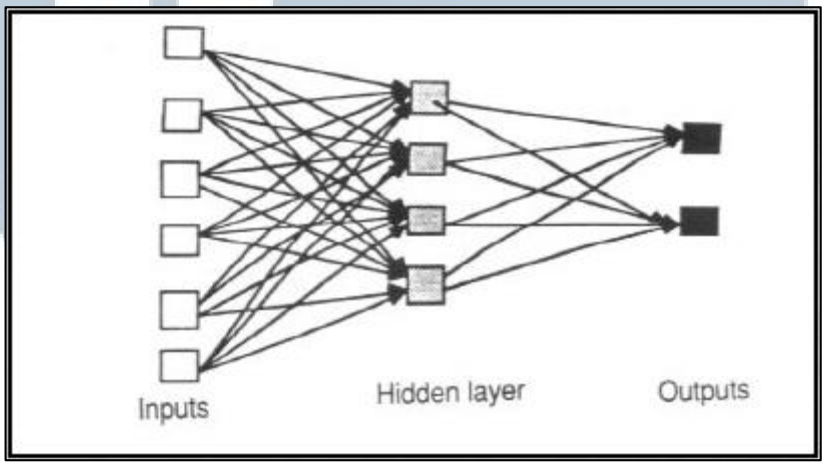
Pembelajaran mesin atau *machine learning* adalah perkembangan dari bidang kecerdasan buatan yang terpusat pada kemampuan mesin meniru kecerdasan manusia. Sesuai namanya, pembelajaran mesin berfokus pada bagaimana membuat suatu mesin dapat mempelajari informasi secara inferensi induktif, yaitu dengan memperhatikan contoh-contoh yang ada dan kemudian menarik kesimpulan (Rätsch, 2004). Dalam bidang pembelajaran mesin, apabila setiap contoh pada set pelatihan mewakili sebuah jawaban yang diskrit (disebut *supervised learning*), maka dapat dikategorikan sebagai masalah klasifikasi pola atau *pattern recognition*. Setelah mesin mempelajari contoh-contoh yang diberikan, mesin diharapkan dapat menyimpulkan jawaban untuk kasus baru yang belum pernah diobservasi sebelumnya (Rätsch, 2004).

2.5 Jaringan Saraf Tiruan

Jaringan saraf tiruan adalah paradigma pemrosesan informasi yang terinspirasi dari cara kerja jaringan saraf biologis. Pada dasarnya, jaringan saraf tiruan adalah sebuah *graph* yang terdiri atas elemen-elemen pemrosesan (disebut unit neuron) yang saling berhubungan dan digunakan untuk menyelesaikan masalah spesifik, seperti pengenalan pola dan klasifikasi data. Setiap neuron memiliki nilai numerik pada titik dan jalur transformasinya dan nilai tersebut berubah sesuai

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

dengan contoh-contoh masukan yang digunakan untuk melatih jaringan (Stergiou & Siganos, 1996). Arsitektur jaringan saraf tiruan bekerja sebagai pemroses statistik, dan melakukan asumsi probabilistik dari data masukan. Berdasarkan dari sejumlah data pelatihan, jaringan saraf tiruan diharapkan dapat menghasilkan sebuah model statistik dari kumpulan data tersebut, sehingga dapat membuat prediksi yang terbaik untuk data yang baru (Jordan & Bishop, 1996). Struktur dasar jaringan saraf tiruan dapat dilihat pada Gambar 2.2.



Gambar 2.2 Struktur Jaringan Saraf Tiruan *Feed-forward* Sederhana (Stergiou & Siganos, 1996)

Jaringan saraf tiruan pada umumnya terdiri atas tiga layar, yaitu *input* atau masukan, *hidden layer* atau layar tersembunyi, dan *output* atau keluaran. Satu buah layar tersembunyi sudah cukup agar sistem dapat mengenali masukan, namun penambahan jumlah layar tersembunyi dapat memudahkan proses pelatihan (Siang, 2005). Aktivitas neuron pada layar masukan merepresentasikan informasi yang dimasukkan ke dalam jaringan, aktivitas neuron pada layar tersembunyi ditentukan oleh aktivitas neuron layar masukan dan bobot dari jalur koneksi antara neuron masukan dan neuron tersembunyi, dan aktivitas neuron pada layar keluaran

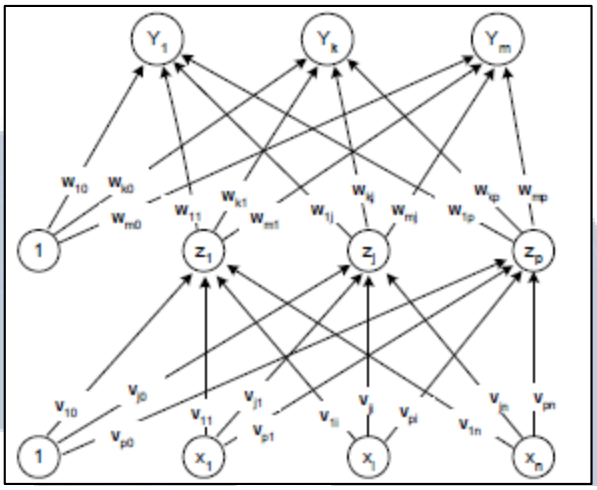
Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

ditentukan oleh aktivitas neuron layar tersembunyi dan bobot dari jalur koneksi antara neuron tersembunyi dan neuron keluaran (Stergiou & Siganos, 1996).

Jaringan saraf tiruan menyimpan informasi dari suatu pola dengan mengeluarkan respons sesuai dengan sifat khusus dari masukan. Setiap neuron dari jaringan memiliki respon masing-masing terhadap suatu masukan yang mengandung pengartian khusus. Pengetahuan jaringan saraf tiruan terhadap informasi terletak pada nilai dari bobot koneksi, dan proses pelatihan atau pembelajaran suatu jaringan saraf tiruan adalah proses penentuan nilai dari bobot tersebut (Stergiou & Siganos, 1996). Perilaku dari suatu jaringan saraf tiruan bergantung pada nilai bobot-bobot tersebut dan fungsi transfer atau fungsi *input-output* yang digunakan pada unit neuron. Fungsi aktivasi yang sering digunakan adalah fungsi sigmoid biner yang menghasilkan nilai keluaran 0 sampai 1, dan sigmoid bipolar yang menghasilkan nilai keluaran -1 sampai 1 (Siang, 2005).

Jaringan diinisialisasi dengan mengisi matriks bobot dengan nilai bobot acak dengan besar -0.5 sampai dengan 0.5. Untuk melatih jaringan saraf tiruan sehingga perilaku jaringan semakin mendekati perilaku yang diinginkan, maka nilai bobot harus diubah sehingga nilai eror antara keluaran yang diinginkan dan keluaran sebenarnya berkurang. Proses ini dilakukan dengan cara menghitung nilai derivatif dari eror pada bobot, dan memperhatikan perubahan dari nilai eror tersebut ketika nilai bobot diubah sedikit demi sedikit. Algoritma yang umum digunakan untuk menghitung nilai derivatif eror tersebut adalah *backpropagation* (Stergiou & Siganos, 1996).

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 2.3 *Backpropagation* pada Jaringan Saraf Tiruan dengan Satu Layar Tersembunyi (Siang, 2005)

Sesuai dengan jaringan Gambar 2.3, proses perhitungan *backpropagation* dimulai dengan menghitung nilai keluaran sistem dari nilai masukan dalam fase propagasi maju. Hal ini dilakukan dengan menghitung nilai unit layar tersembunyi (z_net_j) dengan mengalikan layar masukan (x_i) dengan matriks bobotnya (v_{ji}). Nilai unit layar tersembunyi kemudian dimasukkan ke dalam fungsi aktivasi yang digunakan (Siang, 2005). Nilai tersebut kemudian dapat diteruskan dan dikalikan terhadap matriks bobot (w_{kj}) untuk mendapatkan nilai layar keluaran (y_net_k).

Setelah mendapatkan besar vektor layar keluaran, nilai-nilai elemen pada vektor keluaran (y_k) dapat diselisih terhadap nilai-nilai elemen pada vektor target yang harus dicapai (t_k) untuk mendapatkan nilai faktor kesalahan pada layar keluaran (δ_k) dalam fase propagasi mundur (Siang, 2005). Dalam bentuk vektor kolom, perhitungan besar vektor faktor kesalahan (δ_y) dapat dihitung berdasarkan selisih antara vektor target (T) dan vektor keluaran (Y).

Berdasarkan vektor kesalahan layar keluaran (δ_y), dapat dihitung besar vektor kesalahan pada matriks bobot (δ_w) dan pada layar berikutnya (δ_z) (Mallya, A Quick Introduction to Backpropagation, 2015). Besar vektor pada layar tersembunyi Z

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

kemudian digunakan untuk menghitung besar vektor kesalahan pada matriks bobot (δ_v). Fase propagasi mundur selesai sampai pada layar masukan (X) (Siang, 2005).

Setelah seluruh besar vektor kesalahan didapatkan, maka fase berikutnya adalah mengubah nilai bobot. Untuk meminimalisir nilai eror untuk masukan sistem berikutnya, maka digunakan fungsi *stochastic gradient descent* (SGD) (Minnaar, 2015). Nilai perubahan bobot pada matriks bobot dapat dihitung menggunakan perkalian laju pemahaman (α) dengan nilai elemen vektor kesalahan matriks tersebut. Nilai perubahan bobot tersebut kemudian dijumlahkan dengan nilai bobot sebelumnya untuk digunakan dalam pemrosesan masukan berikutnya (Siang, 2005).

Proses propagasi mundur dilakukan terus selama proses pelatihan jaringan untuk seluruh *dataset* pelatihan. Dalam suatu pengujian sistem jaringan saraf tiruan, (Shahin, Maier, & Jaksa, 2004) menyatakan bahwa 20% dari data masukan digunakan untuk set validasi, sedangkan sisanya dibagi sebanyak 70% untuk pelatihan dan 30% untuk pengujian. Set pelatihan digunakan untuk propagasi mundur dan pelatihan jaringan dengan mengubah nilai matriks bobot, set validasi digunakan untuk menghindari terjadinya *overfitting*, dan set pengujian digunakan untuk menghitung akurasi akhir dari sistem.

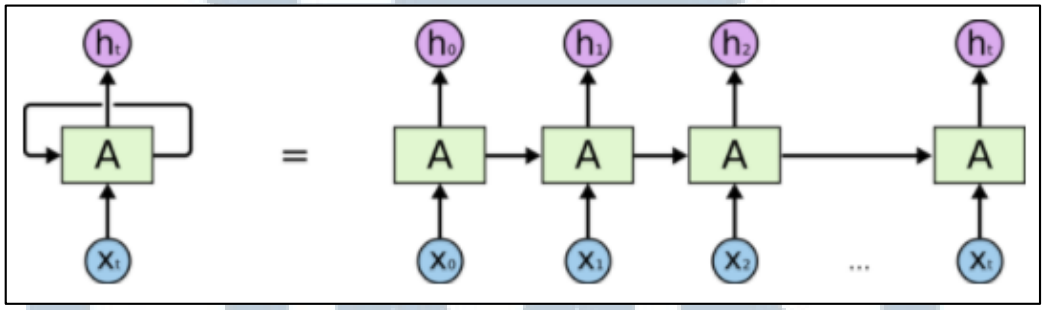
Pelatihan sistem dilakukan dalam bentuk pengulangan yang disebut *epoch*. Dalam satu buah *epoch*, sistem dimasukkan dengan seluruh data dalam set pelatihan, dan menyesuaikan nilai bobot matriks untuk setiap tahap propagasi mundur dari seluruh data. Untuk setiap 5 kali pengulangan *epoch*, (Prechelt, 1997) menganjurkan untuk menghitung nilai eror dari set validasi. Apabila nilai eror set validasi bertambah dari hasil perhitungan sebelumnya, atau setelah 3000 *epoch* telah dilakukan, maka pelatihan dapat dihentikan. Sistem kemudian diuji

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

menggunakan data dari set pengujian dan dihitung nilai akurasinya (Shahin, Maier, & Jaksa, 2004).

2.6 Recurrent Neural Network

Recurrent Neural Network (RNN) adalah sebuah model jaringan saraf tiruan yang cocok digunakan untuk klasifikasi pola yang memiliki masukan berupa urutan (Sutskever, 2013). RNN menggunakan jaringan yang memiliki perulangan pada layar tersembunyi di dalamnya, dan mengambil masukan dari data masa kini dan masa lampau (Skymind, 2017). Struktur RNN, seperti yang ditunjukkan pada Gambar 2.4, memiliki relasi antar unit pada layar tersembunyi yang sama yang berfungsi untuk menyampaikan informasi dari masukan waktu sebelumnya untuk memengaruhi keputusan berikutnya (Olah, 2015).



Gambar 2.4 Struktur Perulangan RNN Terhadap Waktu (Olah, 2015)

Menggunakan perulangan tersebut, RNN memiliki kelebihan dibandingkan dengan model jaringan saraf tiruan lainnya, yaitu pada kemampuannya untuk menyimpan informasi dari masukan-masukan pada masa lampau untuk masukan berikutnya (Olah, 2015). Namun, struktur RNN yang memiliki jaringan perulangan dan nonlinier mengakibatkan kesulitan dalam tahap pelatihan, terutama pada kondisi ketergantungan waktu yang lama (Sutskever, 2013). Perubahan nilai bobot yang dihitung pada tahap *backpropagation* berubah secara eksponensial seiring

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

dengan propagasinya, sehingga nilai tersebut, sekecil apapun, dapat dengan cepat menghilang (*vanishing gradient*) atau membesar (*exploding gradient*) (Gers, 2001).

2.7 Long Short-Term Memory

Long Short-Term Memory (LSTM) adalah varian dari RNN yang dapat mempertahankan nilai eror ketika proses *backpropagation* dilakukan melalui waktu dan layar, memungkinkan RNN untuk menyimpan informasi lebih dari 1000 langkah waktu tanpa mengalami permasalahan *vanishing gradient* (Skymind, 2017). LSTM memiliki sebuah jalur penyimpanan informasi tambahan di luar dari aliran informasi RNN yang disebut *gated cell*. Informasi yang melalui jalur ini hanya mengalami perubahan linier sehingga nilai yang terkandung tidak mudah berubah. Perubahan terhadap informasi pada sel hanya dapat dilakukan melalui struktur gerbang yang disebut *gates* (Olah, 2015).

Gerbang terdiri atas unit sigmoid dan operasi perkalian titik atau *pointwise*, dengan nilai keluaran antara 0 sampai dengan 1. Nilai 0 melambangkan gerbang yang tertutup, sedangkan nilai 1 melambangkan gerbang yang terbuka (Olah, 2015). Informasi dapat dikirim, dibaca, dan dihapuskan melalui struktur gerbang yang membuka dan menutup sesuai dengan nilai sigmoid tersebut. Nilai sigmoid gerbang ditentukan oleh informasi dan nilai bobot gerbang tersebut. Nilai bobot pada gerbang juga disesuaikan melalui proses pembelajaran *backpropagation*, sehingga gerbang mengetahui kondisi untuk mengirim, menyimpan, atau membaca informasi melalui proses iterasi pelatihan (Skymind, 2017).

Terdapat tiga gerbang yang mengatur alur informasi dalam sel LSTM (Olah, 2015). Gerbang pertama yang dilalui adalah *forget gate*. Gerbang ini menentukan apakah informasi dari sel sebelumnya sebaiknya dihapuskan atau disimpan.

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

Menurut (Sutskever, 2013), perlu ditambahkan nilai bias eksplisit (b_f) yang tinggi (misalkan 5) pada gerbang ini agar gerbang mendapatkan nilai mendekati 1 pada tahap awal pembelajaran untuk menghindari terjadinya *vanishing gradient*. Nilai keluaran gerbang *forget* (f_t) ditentukan oleh Rumus 2.1 (Mallya, LSTM Forward and Backward Pass, 2015).

$$f_t = \sigma(\hat{f}_t + bf) = (W_f X_t + U_f h_{t-1} + bf) \quad \ldots(2.1)$$

Gerbang kedua adalah *input gate*, yang menentukan informasi yang diubah atau ditambahkan ke dalam sel. Nilai keluaran gerbang *input* (i_i), nilai masukan (a_t), dan nilai sel memori (c_t) ditentukan oleh Rumus 2.2, 2.3, dan 2.4 secara berurutan.

$$i_t = \sigma(\hat{\imath}_t) = \sigma(W_i X_t + U_i h_{t-1}) \quad \ldots(2.2)$$

$$a_t = \tanh(\hat{a}_t) = \tanh(W_c X_t + U_c h_{t-1}) \quad \ldots(2.3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot a_t \quad \ldots(2.4)$$

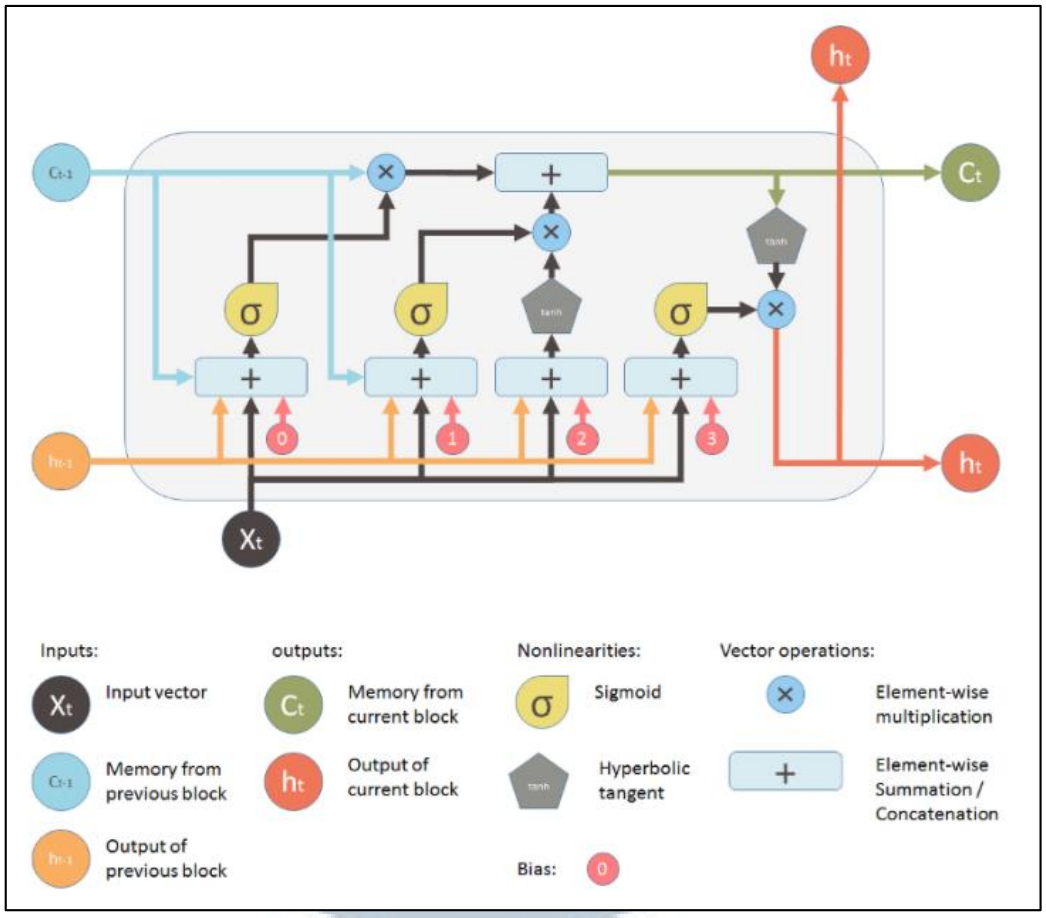
Gerbang ketiga adalah *output gate*, yang menentukan informasi yang digunakan sebagai hasil keluaran (Olah, 2015). Nilai keluaran gerbang *output* (o_t) dan nilai keluaran satu putaran (h_t) ditentukan oleh Rumus 2.5 dan Rumus 2.6.

$$o_t = \sigma(\hat{o}_t) = \sigma(W_o X_t + U_o h_{t-1}) \quad \ldots(2.5)$$

$$h_t = o_t \odot \tanh(c_t) \quad \ldots(2.6)$$

Untuk menggunakan representasi spektrogram sebagai masukan ke dalam jaringan LSTM, maka spektrogram dapat dianggap sebagai urutan vektor-vektor kolom terhadap waktu yang terdiri atas sejumlah elemen sesuai dengan tinggi gambar spektrogram. Masing-masing dari elemen memiliki nilai sesuai dengan intensitas warna yang ada pada sel tersebut.

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 2.5 Ilustrasi Struktur Modul Perulangan LSTM dan Operasi Vektornya (Santos, 2017)

Untuk memetakan hasil perhitungan dalam unit LSTM menjadi hasil akhir, maka keluaran tersebut pada tahap akhir LSTM perlu diubah menjadi keluaran berupa nilai kepercayaan terhadap setiap klasifikasi yang ada. *Softmax classifier* adalah fungsi klasifikasi yang dapat menangani jumlah klasifikasi yang tidak tetap. Fungsi ini memberikan nilai probabilitas dari nilai skor seluruh klasifikasi label untuk suatu masukan (UFLDL, 2013). Contoh proses *softmax classifier* dijelaskan pada Gambar 2.6.

UNIVERSITAS MULTIMEDIA NUSANTARA

	Scoring Function	Unnormalized Probabilities	Normalized Probabilities
Dog	-3.44	0.0321	0.0006
Cat	1.16	3.1899	0.0596
Boat	-0.81	0.4449	0.0083
Airplane	3.91	49.8990	0.9315

Gambar 2.6 Contoh *Softmax Classifier* Memberikan Nilai Probabilitas untuk Klasifikasi (Rosebrock, 2016)

Sebagai layar keluaran pada jaringan saraf tiruan, *softmax classifier* menerima masukan berupa vektor dari layar tersembunyi yang dipetakan terhadap klasifikasi keluaran melalui layar *fully-connected,* yaitu perkalian titik antara matriks bobot W dengan vektor masukan x (UFLDL, 2013). Vektor yang dihasilkan perkalian Rumus 2.7 disebut sebagai skor (s).

$$s_t = W_s \times h_t \quad \ldots(2.7)$$

Vektor skor yang didapatkan kemudian diubah menjadi vektor yang berisikan nilai positif (p_t) pada Rumus 2.8, dengan panjang vektor sesuai dengan jumlah klasifikasi yang ada melalui layar *softmax*. Elemen dengan nilai yang tinggi memiliki nilai positif yang lebih besar. Nilai positif yang dihasilkan digunakan untuk menghasilkan nilai kepercayaan (P_t) melalui Rumus 2.9 untuk masing-masing klasifikasi. Nilai total dari seluruh nilai kepercayaan setiap klasifikasi akan berjumlah 1.

$$p_t = e^{s_t} \quad \ldots(2.8)$$

$$P_t = \frac{p_t}{\sum_j e^{s_j}} \quad \ldots(2.9)$$

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

Melalui nilai kepercayaan masing-masing klasifikasi yang dikeluarkan, nilai kesalahan atau eror dari suatu masukan (E_i) dapat dihitung. Perhitungan nilai kesalahan dari seluruh *dataset* (E) pada rumus 2.11 kemudian dapat dilakukan dengan menghitung rata-rata dari seluruh nilai eror masukan (E_i) pada Rumus 2.10. Nilai ini kemudian dapat digunakan dalam menentukan kapan *epoch* perlu dihentikan lebih awal.

$$E_i = -ln(P_t) \quad \ldots(2.10)$$

$$E = \frac{1}{n} \sum_{i=1}^{n} E_i \quad \ldots(2.11)$$

Nilai kepercayaan yang dihasilkan merupakan nilai keluaran sistem terhadap suatu masukan, dan digunakan pada tahap *backpropagation*. Faktor kesalahan keluaran pada vektor skor (δs_t) dapat dihitung menggunakan nilai dari vektor kepercayaan (P_t) dan vektor target (t) pada Rumus 2.12 (Mallya, A Quick Introduction to Backpropagation, 2015).

$$J_{ij} = P_{t_i} \odot (1\{i = j\} - P_{t\,j}) \quad \ldots(2.12)$$

$$\delta_{s_t} = J \times (P_t - t) \quad \ldots(2.13)$$

Menggunakan nilai faktor kesalahan (δs_t), faktor kesalahan pada layar *fully connected* yaitu faktor kesalahan matriks bobot (δW_s) dan faktor kesalahan vektor keluaran satu putaran (δh_t) dapat dihitung melalui Rumus 2.13 dan Rumus 2.14.

$$\delta_{W_s} = \delta_{s_t} \times {h_t}^T \quad \ldots(2.14)$$

$$\delta_{h_t} = {W_s}^T \times \delta_{s_t} \quad \ldots(2.15)$$

Karena layar tersembunyi LSTM terdiri atas beberapa matriks bobot yang terpisah-pisah, maka dapat dijabarkan pada Rumus 2.15 dan Rumus 2.16 sebuah vektor layar tersembunyi (z_t) yang merupakan hasil perkalian antara matriks

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

keseluruhan (W) dengan vektor masukan pada tahap tersebut (I_t) yang terdiri atas vektor masukan (x_t) dan vektor keluaran LSTM tahap sebelumnya (h_{t-1}).

$$z_t = \begin{bmatrix} \hat{a}_t \\ \hat{\imath}_t \\ \hat{f}_t \\ \hat{o}_t \end{bmatrix} = \begin{bmatrix} W_c & U_c \\ W_i & U_i \\ W_f & U_f \\ W_o & U_o \end{bmatrix} \times \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \quad \ldots(2.16)$$

$$z_t = W \times I_t \quad \ldots(2.17)$$

Menggunakan nilai vektor kesalahan keluaran *softmax* akhir (δh_t), maka dapat dihitung besar vektor kesalahan pada gerbang keluaran (δo_t) menggunakan Rumus 2.17 dan sel memori (δc_t) menggunakan Rumus 2.18. Vektor kesalahan pada sel memori (δc_t) merupakan penjumlahan dari seluruh vektor (δc_t) yang sudah dihitung sebelumnya (Mallya, LSTM Forward and Backward Pass, 2015).

$$\delta o_t = \delta h_t \odot \tanh(c_t) \quad \ldots(2.18)$$

$$\delta c_t \mathrel{+}= \delta h_t \odot o_t \odot (1 - \tanh^2(c_t)) \quad \ldots(2.19)$$

Vektor kesalahan pada sel memori (δc_t) kemudian dapat digunakan untuk menghitung vektor kesalahan pada gerbang masukan (δi_t), nilai masukan (δa_t), gerbang *forget* (δf_t), dan sel memori tahap sebelumnya (δc_{t-1}) pada Rumus 2.19, Rumus 2.20, Rumus 2.21, dan Rumus 2.22 secara berurutan.

$$\delta i_t = \delta c_t \odot a_t \quad \ldots(2.20)$$

$$\delta a_t = \delta c_t \odot i_t \quad \ldots(2.21)$$

$$\delta f_t = \delta c_t \odot c_{t-1} \quad \ldots(2.22)$$

$$\delta c_{t-1} = \delta c_t \odot f_t \quad \ldots(2.23)$$

Vektor-vektor tersebut kemudian digunakan untuk menghitung vektor kesalahan layar tersembunyi (δz_t) melalui Rumus 2.23 sampai dengan Rumus 2.27.

$$\delta \hat{a}_t = \delta a_t \odot (1 - \tanh^2(\hat{a}_t)) \quad \ldots(2.24)$$

$$\delta \hat{\imath}_t = \delta i_t \odot i_t \odot (1 - i_t) \quad \ldots(2.25)$$

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

$$\delta \hat{f}_t = \delta f_t \odot f_t \odot (1 - f_t) \quad \ldots(2.26)$$

$$\delta \hat{o}_t = \delta o_t \odot o_t \odot (1 - o_t) \quad \ldots(2.27)$$

$$\delta z_t = [\delta \hat{a}_t \quad \delta \hat{\imath}_t \quad \delta \hat{f}_t \quad \delta \hat{o}_t]^T \quad \ldots(2.28)$$

Menggunakan vektor (δz_t), dapat dihitung vektor kesalahan masukan tahap (δI_t) pada Rumus 2.28 dan matriks bobot keseluruhan (δW_t) pada Rumus 2.29. Vektor kesalahan (δI_t) seperti pada Rumus 2.28 kemudian dapat dipisah untuk mendapatkan vektor kesalahan keluaran layar LSTM tahap sebelumnya (δh_{t-1}).

$$\delta I_t = \begin{bmatrix} \delta x_t \\ \delta h_{t-1} \end{bmatrix} = W^T \times \delta z_t \quad \ldots(2.29)$$

$$\delta W_t = \delta z_t \times (I_t)^T \quad \ldots(2.30)$$

Setelah melakukan propagasi mundur terhadap seluruh iterasi LSTM untuk satu masukan, maka vektor kesalahan matriks bobot keseluruhan LSTM (δW) dapat dihitung seperti pada Rumus 2.30. Matriks bobot LSTM dan skor kemudian diubah sesuai dengan *stochastic gradient descent* dari nilai eror yang telah didapatkan pada Rumus 2.31 dan Rumus 2.32.

$$\delta W = \sum_{t=1}^{T} \delta W_t \quad \ldots(2.31)$$

$$W = W + \Delta W = W + \alpha \delta W \quad \ldots(2.32)$$

$$W_s = W_s + \Delta W_s = W_s + \alpha \delta W_s \quad \ldots(2.33)$$

2.8 NumPy

NumPy (Numeric Python) adalah sekumpulan ekstensi dari bahasa pemrograman Python yang memiliki kemampuan pengolahan *array* multidimensi yang berukuran besar secara efisien (Ascher, Dubois, Hinsen, Hugunin, & Oliphant, 1999). NumPy mendukung pengolahan *array* multidimensi dan matriks serta memiliki banyak fungsi-fungsi matematis untuk *array* dan matriks tersebut.

Implementasi Long Short-Term..., Adrian Hartanto Tedja, FTI UMN, 2018

NumPy menawarkan kemampuan pengolahan angka dalam jumlah yang besar dengan kecepatan yang tinggi dan alokasi memori yang kecil. NumPy merupakan *library* Python yang fundamental dalam mempersiapkan lingkungan pemrograman untuk perhitungan ilmiah (Oliphant, 2006).

2.9 SoX

SoX (Sound eXchange) adalah alat pemrosesan sinyal suara berbasis *command line* yang cocok digunakan untuk melakukan manipulasi sinyal suara yang cepat, sederhana, dan dalam jumlah yang besar (Bagwell, 2014). SoX dapat membaca dan menyimpan *file* suara dengan jenis *format* yang populer dan memberikan modifikasi terhadap suara tersebut, seperti efek, *combine*, sintesis suara, *split*, dan lainnya. SoX juga dapat membentuk keluaran spektrogram dari suatu *file* suara dalam bentuk *Portable Network Graphic* (PNG) (Bagwell, 2014). Waktu ditunjukkan pada sumbu x, frekuensi pada sumbu y, dan intensitas suara pada sumbu z menggunakan representasi warna *pixel*.

UNIVERSITAS MULTIMEDIA NUSANTARA